Lesson 5

Understanding Desktop Applications

Knowledge Assessment

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You need to design a Windows service that cannot be paused. Which of the following options will help you accomplish this task?
   - **a. Set the CanPauseAndContinue property of the Windows service to False.**
   - b. Set the CanPauseAndContinue property of the Windows service to True.
   - c. Set the CanStart property of the Windows service to True, but set the CanShutdown property to False.
   - d. Do not override the OnPause and OnContinue methods in the Windows service.
2. You have developed a Windows service and need to install this service in order to install its functionality. Which of the following options should you choose to accomplish this task?
   - a. Use the Visual Studio Server Explorer.
   - b. User the Services node in the Computer Management window.
   - **c. Use InstallUtil.exe.**
   - d. Use gacutil.exe.
3. You have developed a Windows service. This service needs to run as a nonprivileged user in order to minimize any possible security risk. Which of the following accounts should you use for running this Windows service?
   - a. LocalSystem
   - b. NetworkService
   - **c. LocalService**
   - d. User (where the UserName property is set to a member of administrator role)
4. You are designing a Windows service application that contains only one Windows service. You would like this service to be started automatically when the computer is restarted. Which of the following classes should you use to specify this setting?
   - a. ServiceBase
   - **b. ServiceInstaller**
   - c. ServiceProcessInstaller
   - d. ServiceController

5. You need to change the display and behavior of a Windows Form so that the form can contain multiple child windows. What should you do?
   a. **Set the IsMdiContainer property of the form to True.**
   b. Set the MdiParent property for all the child windows.
   c. Set the MdiChild property of the form.
   d. Set the IsMdiChild property of the form.
6. You are developing a Windows Form that responds to mouse events. When the mouse moves, you need to invoke the method Form1_HandleMouse. Any code that you write should not affect any existing event-handling code. What statement should you use to attach the event handler to the event?

   a.

   this.MouseDown = new MouseEventHandler

   (Form1_HandleMouse);

   b.

   this.MouseMove = new MouseEventHandler

   (Form1_HandleMouse);

   c.

   this.MouseDown += new MouseEventHandler

   (Form1_HandleMouse);

   d.

   this.MouseMove += new MouseEventHandler

   (Form1_HandleMouse);

7. You are developing a Windows form with a multiple document interface (MDI). You need to write code that arranges the child windows vertically within the client region of the MDI parent form. Which of the following statements should you use?

   a.

   LayoutMdi(MdiLayout.TileVertical);

   b.

   LayoutMdi(MdiLayout.Cascade);

   c.

   MdiLayout(LayoutMdi.TileVertical);

   d.

   MdiLayout(LayoutMdi.Cascade);

8. You are developing an application that will be run from the command line. Which of the following methods would you use for output to the command line?

   a. Console.Read
   b. Console.Write
   c. File.Read
   d. File.Write

9. You want to develop an application that displays a visual surface capable of displaying a variety of controls, such as text boxes, buttons, and menus. The application should also allow multiple child windows to reside under a single parent window. Which of the following types of applications should you develop?
   a. Console-based application.
   b. Windows service application.
   c. Single document interface (SDI) application.
   d. Multiple document interface (MDI) application.

10. You are extending an existing Windows application. You would like to create a new form that derives its visual characteristics (including size, color, and some controls) from a previously created form. Which technique should you use to create the new form?
   a. Visual inheritance.
   b. Visual encapsulation.
   c. Visual abstraction.
   d. Visual polymorphism.

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. Use the **Description** property of the ServiceInstaller class to specify a brief comment that explains the purpose of the service.
2. The **Account** property of the **ServiceProcessInstaller** class indicates the account type under which a Windows service will run.
3. The **Source** property of the EventLog class is use to specify the application name to use when writing to an event log.
4. **Visual inheritance** allows you to reuse existing functionality and layout for Windows Forms.
5. **Multiple document interface (MDI)** applications are applications in which multiple child windows reside under a single parent window.
6. A(n) **Windows service** is ideal for creating long-running applications that run in the background and do not have any user interface.
7. **Console-based applications** do not have a graphical user interface and use a text-mode console window to interact with the user.

8. **Multiple document interface (MDI)** applications provide their own window management functionality, whereas **single document interface (SDI)** applications rely on the operating system for window management.
9. A delegate can be bound to any method whose signature matches that of the **delegate declaration**.
10. The **multicast delegates** can be bound to more than one method, allowing one-to-many notifications when an event is fired.

Competency Assessment

Project 5-1: Using Visual Inheritance

You need to create a Windows Form similar to the one you created in the VisualInheritance exercise. However, this time, the requirement is that the background color of this form must match the currently selected color of the user's desktop. How would you develop such a form?

1. **Open the Windows Application project named WindowsFormsDesign.**
2. **Add a new Windows Form based on the Inherited Form template. Name the inherited form BackColor.cs.**
3. **In the Inheritance Picker dialog box, select InheritedForm.**
4. **In the Designer view of the BackColor form, set the Text property to "BackColor Form."**
5. **In the Designer view, select the BackColor form. From the Properties window, set the BackColor property to Desktop. Set the ForeColor property to HighlightText.**
6. **Open Program.cs and modify the Main method as shown below to make sure that BackColor form is launched when you run the application:**

```
[STAThread]
static void Main()
{
  Application.EnableVisualStyles();
  Application
   .SetCompatibleTextRenderingDefault(false);
  Application.Run(new BackColor());
}
```

7. **Build and run the project.**

Project 5-2: Handling the MouseDown Event

You are developing a game that allows users to hit a target area on a Windows Form with their mouse. You need to develop an experimental form that displays the X and Y coordinates of the location clicked by the user in the form's title bar. How should you achieve this?

1. **Create a new Windows Forms project with the name MouseDown.**
2. **In the Properties window of the form, click the Event icon and look for an event named MouseDown. Double-click the row containing the MouseDown event. Modify the MouseDown event handler created by Visual Studio to the following:**

```
private void Form1_MouseDown(object sender,
    MouseEventArgs e)
{
  Text = String.Format(
   "X = {0}, Y = {1}", e.X, e.Y);
}
```

3. **Build and run the project. Click on the client area of the form and notice that the text in the title bar shows the coordinates of the point at which the mouse button was pressed.**

Proficiency Assessment

Project 5-3: Working with Console Input

You are developing a program that manipulates text. You need to write a console-based application that accepts text from the user and converts the text to upper-case letters. What code do you need to write to meet this requirement?

1. **Create a new ConsoleApplication and name it ToUpper.**
2. **Add the following code to the Main method:**

```
static void Main(string[] args)
{
  int character = Console.Read();

  while (character != -1)
  {
    if (Char.IsLetter((char)character))
      Console.Write(((char)character)
            .ToString().ToUpper());
        else
            Console.Write( (char) character);

        character = Console.Read();
    }
}
```

3. **Build and run the application. Type some letters on the command line and press Enter. You should see that the letters are converted to upper case. Press Ctrl+Z (end of input) to end the program.**
4. **Copy DisplayFile.exe from the previous exercise and ToUpper.exe to the same folder. Run the following command from the command line:**

```
DisplayFile Sample.txt | ToUpper
```

Notice that the generated output is the contents of the Sample.txt file converted to uppercase letters. The '|' character works as a pipe between commands, so the console output from the DisplayFile.exe works as console input to the ToUpper.exe. By using pipes, you can chain multiple simple commands to accomplish complicated tasks.

Project 5-4: Using the Net Utility (net.exe)

The net.exe command line utility comes installed with Windows. This utility allows you to perform various networking commands, including control of Windows services. Say you want to use net.exe to work with the previously created FirstService Windows service. What steps must you take in order to pause, stop, and start a Windows service using the net.exe utility?

1. **Open the command prompt. To enumerate a list of all installed services, issue the following command:**

```
net start
```

2. **To start the FirstService Windows service, give the following command:**

```
net start FirstService
```

3. **To pause FirstService, give the following command:**

```
net pause FirstService
```

4. **To resume the paused FirstService, give the following command:**

```
net continue FirstService
```

5. To stop FirstService, give the following command:

```
net stop FirstService
```

6. Check the Application event log to see the messages generated by FirstService.